Python

Functions and Lambdas

@kabirbaidhya

Reflections

What we already know

From the last session.

1. Data Types in Python
2. Conditionals and Loops

Note: If you're not aware of these. Read them at https://github.com/kabirbaidhya/learn-python-django-web

Functions

Functions

A function is a block of code that performs a single action or performs some computation.

Usually functions take in some input, process it and returns the output.

Using Functions

In programming use of functions do provide:

- Modularity
- Reusability
- Maintainability
- Separation of Concern
- Better Code Organization

Functions in Python

In python we can define a function with the following syntax:

```
def function_name():
    STATEMENTS
```

If the function needs to accept parameters then it will have following syntax:

```
def function_name(parameters):
    STATEMENTS
```

Example 1

This could be an example of a function that computes sum of two numbers.

```
def sum(x, y):
    return x + y
```

Example 2

And similarly this could be an example of a function which returns the maximum in between two numbers.

```
def max(a, b):
    if a > b:
        return a

    return b

# Print maximum value
print max(5, 6)
```

Example 3

The earlier example can be done using the inline conditional operator like this:

```
def max(a, b):
    return a if a > b else b

# Print maximum value
print max(5, 6)
```

Example 4

Function to print the fibonacci series.

```
def fib(n):
    a, b = 0, 1

    while a < n:
        print(a)
        (a, b) = (b, a + b)

# Print the series
print fib(50)
```

Don't Forget

1. The colon `:`, which actually starts the function block.
2. Indentation, which is the part of syntax in python and a must have (unlike C-style languages where it's optional).
3. End of indent means end of the block.

Naming Convention

In python it's a community wide convention to use `snake_case` (i.e lowercase with underscores to separate words) for naming functions and variables.

For example these could be some examples of good function names:

```
get_user_name()
create_user()
add_user()
validate()
```

Default Arguments

It's possible to set default arguments in python by assignning default values to parameters.

They also act as optional parameters as the user can omit them.

```
def greet(message = 'Hello'):
    return message + ' World!'

# Invoke this function
print greet()        # Hello World!
print greet('Hi')    # Hi World!
```

Function Invocation

Calling/invoking the function is as simple as

```
sum(5, 6)
```

Do Remember:

- Parentheses are the must haves, both during definition and invocation.
- All parameters are passed by reference in Python.

Positional Arguments

Calling a function with positional argument means calling it by providing the arguments in the order they're defined as parameters in the function.

For instance:

```
def sum(x, y, z = 0):
    return x + y + z


sum(1, 2, 3)          # 3 positional arguments
sum(1, 5)             # 2 positional arguments for x & y, z
```

Keyword Arguments

Functions can also be called using keyword arguments using the parameter names in the form `arg=value`.

For instance:

```
def sum(x, y, z = 0):
    return x + y + z

sum(x=1, y=2, z=3)          # keyword arguments
sum(y=5, z=1, x=4)          # position of args doesn't matte
sum(x=1, y=5)               # 2 keyword arguments for x & y,
```

Combine both positional and keyword arguments

You can use both the positional and keyword arguments like this:

```
sum(1, y=3, z=5)
sum(5, 6, z=10)
sum(10, y=2)
```

But ensure that keyword arguments do follow positional arguments and same arguments aren't provided more than once.

The `return` statement

Returns a value from inside the function to the caller of the function.

For example:

```
def foo():
    # This function returns the value 'Bar'
    return 'Bar'


def baz():
    # This function returns nothing
    return

a = foo()       # a = 'Bar'
b = baz()       # b = None
```

Using the `main()` function

Using the `main()` function

Although python gives you full flexibility to structure your code as you like. It is recommended that you make your code organized and modular.

In many programming languages they strictly enforce you to define a `main()` function and don't allow to write code outside a function.
a
Python doesn't force you to do the same, but it's a good practice to have something like `main()` function to keep your code organized.

For Example

```
def fib(n):
    a, b = 0, 1

    while a < n:
        print(a)
        (a, b) = (b, a + b)


def main():
    n = int(input('N = '))

    # Print the series upto n
    print fib(n)


main()
```

Lambdas

Lambda Expressions

A lambda is a short anonymous function that can be used in expressions.

For example: Lamda expression for computing the product of two numbers

```
c = (lambda a, b: a * b)(2, 5)
```

Or

```
func = lambda a, b: a * b
c = func(2, 5)
```

Use Cases

Lambda functions are pretty handy tool when you make use of a lot of higher order functions and closures. Or need to use functions in expressions or pass them to other functions as arguments.

Especially while doing functional programming, lambdas become pretty common.

Example 5

A very basic use case of lambda functions.

```
# A function that returns another function
# to add a number with another number
def get_adder_of(a = 1):
    return lambda x: x + a

add_five = get_adder_of(5)
add_two = get_adder_of(2)
add_seven = get_adder_of(7)

print('10 + 5 = %d' % add_five(10))
print('8 + 2 = %d' % add_two(8))
print('8 + 7 = %d' % add_seven(8))
```

Example 6

Using map() function and lambda to get a new list of squares of the original list.

```
my_list = [1, 2, 3, 4, 5]

squares = list(map(lambda x: x ** 2, my_list))

# The same thing could be achieved using list comprehensio
squares_2 = [x ** 2 for x in my_list]

print(squares)
print(squares_2)
```

Exercises

Do all the exercises we've done so far in previous lessons using functions to wrap up the logic.

Guidelines

1. Write the primary logic of the program using function(s).
2. Do not invoke the functions directly. Use a `main()` function instead.
3. Protip Read user inputs and print the outputs in the `main()` function keeping just the core logic in the other functions.

Read More?

Links

1. http://www.cs.utah.edu/~germain/PPS/Topics/functions.html
2. https://docs.python.org/3/tutorial/controlflow.html
3. https://www.tutorialspoint.com/python/python_functions.htm
4. http://stackoverflow.com/questions/9450656/positional-argument-v-s-keyword-argument

This slide was a part of course

Python, Django & Web Development

github.com/kabirbaidhya/learn-python-django-web

Thank You

@kabirbaidhya

kabirbaidhya@gmail.com